ArmA 3 Billeting as a Coalition Formation Game

Pearson Garner[1,*], Aaron Lin[1,*], Ruby Harris[1,*], Brent Harrison[1] and Judy Goldsmith[1,*]

[1]University of Kentucky, 329 Rose St. Lexington KY 40506, United States of America

Abstract

The video game *Armed Assault 3* (often shortened to *ArmA 3)* is part of a popular series of online military simulation role-playing games by Bohemia Interactive. Many players take part in cooperative organizations, with larger communities hosting 50–100 players simultaneously on dedicated servers. Prior to hosting a game session, players are assigned to roles corresponding to positions in a rank-based tree structure. Assigning players ("billeting") to the game-specific hierarchy happens *before* play, but billeting can take enormous effort and time to satisfy individual players' preferences over roles, levels of the command hierarchy, and individuals with whom they will interact. A stable assignment of players to their teams is integral both for their enjoyment of the game and the overall effectiveness of the team. Inspired by the existing coalition formation literature, we re-cast that pre-play activity as a *coalition formation game* and explore how various coalition formation algorithms perform on the billeting process using synthetic data.

Keywords

Coalition Formation, Team Assignment, Stable Teams

1. Introduction

The 2013 video game *Armed Assault 3* (usually shortened to ArmA 3) by Bohemia Interactive is a realistic military simulation game, or *MilSim*, which gives players an online sandbox to role-play in military settings. Realistic MilSim communities in ArmA 3 have large organizational structures which emulate real-life chain of command, but these communities often have difficulty assigning players to different in-game roles. This process, called *billeting*, is especially difficult in large MilSim communities because they, especially in ArmA 3, often have hundreds of members, with anywhere from 50 to 100 people trying to play in a cooperative game instance all at once. The first author reports that it takes several days for billeting, and that process often leads to acrimonious disputes, in part driven by interpersonal tensions between players who do or do not wish to interact with each other in fire teams or in adjacent roles in the chain of command. In order to form *stable* teams, i.e., teams where no players have incentive to change their assignment, we define this more formally in the section on coalition formation games. community leadership has to spend a lot of time and energy on team formation, often with mixed success. Given the time and effort required to find these solutions, utilizing algorithmic methods for finding stable teams would greatly reduce the burden on community leadership.

In this paper, we explore how algorithms for forming stable coalitions in coalition formation games can be applied to team formation in ArmA 3. Using stability as a measure of goodness or utility, we examine a number of coalition formation algorithms and evaluate them based on the overall utility (stability) of the billets that they generate. Specifically, we explore how various simulated annealing and local search methods can be used to billet a set of synthetic ArmA 3 users based on a set of known preferences.

The remainder of the paper is organized as follows. In Section 2 we describe related work about coalition formation, and about other games with similar billeting/assignment problems. In Section 3 we provide an overview of both ArmA 3 and the billeting process. We then describe the coalition formation algorithms that we will investigate in this paper and outline our experiments and results.

2. Related Works

We observe billeting problems arising in large-scale online community games, most notably in roleplay settings which require individuals to be divided into teams with a clear structural hierarchy. One example case is for the game EveOnline (https://www.eveonline.com/), in which players and their ships are assigned to fleet formations based on the intrinsic rank and skills of their in-game character. Outside of strictly video games, we note billeting problems being relevant for many other large-community structures, such as coordinating text-based roleplay communities.

A *coalition formation game* is a cooperative multi-game agent game. The input is a set of agents and their preferences or utilities over coalitions they might be assigned to; the output is a partition of agents into coalitions. The computational goal of a coalition game problem might be to find an optimal or near-optimal partition, as we do here, or to find a *stable* partition for some notion of stability, or to determine if such a stable partition exists. We are particularly interested, here, in *hedonic coalition formation games*[1, 2], also known as *hedonic games*, where agents' preferences or utilities are defined only over each coalition they might be assigned to. If we interpret the billeting problem for ArmA 3 as the problem of finding a single (or "grand") coalition with optimal total utility, we do not consider the effects of other teams' billeting, so we are in the hedonic setting.

3. ArmA 3

In this section we describe ArmA 3, and mention the ways in which we codify rules that are left to individual gaming communities.

3.1. Overview

ArmA 3 is a highly diverse community with many different play-styles, game-modes, and levels of realism. For the purpose of this paper, we construct a coalition formation game

11th Experimental Artificial Intelligence in Games Workshop, November 19, 2024, Lexington, Kentucky, USA.

*Corresponding author: Pearson Garner, Aaron Lin, and Judy Goldsmith

pearson.neal.garner@gmail.com (P. Garner); ayli22@uky.edu (A. Lin); roo.harris@proton.me (R. Harris); brent.harrison@uky.edu (B. Harrison); goldsmit@cs.uky.edu (J. Goldsmith)

0009-0007-0375-1650 (P. Garner); 0009-0009-3317-4545 (A. Lin); 0009-0006-0915-8084 (R. Harris); 0000-0002-9421-8566 (B. Harrison); 0000-0002-9421-8566 (J. Goldsmith)

for a “realism"-focused MilSim community.[1] Like many popular communities, we will primarily base our coalition game’s structure off of North American Treaty Organization (NATO) protocol and procedure.

There are no formal rules from the game developers, as each community gets to choose their own organizational rules. Most realistic ArmA 3 units attempt to follow the standard NATO ORBAT that is used by modern Western militaries. (https://web.archive.org/web/20070826233341/http://\orbat.com/site/toe/toe/usa/platoontoe.html) We therefore mimic this structure as well.

3.2. Billeting

The formation of an instance of an ArmA 3 mission (henceforth for simplicity just “mission") first requires each player to be assigned a role for that mission’s “billet" before the mission is hosted at some fixed time. The role a player is assigned is based on several factors, including but not limited to:

1. Player preferences towards working with certain individuals.
2. The player’s rank in the unit.
3. The player’s qualifications earned from training.

The billet is constructed as a tree of players, where each player node has command and authority over every player in each child branch, and has most direct authority over the immediate children to the player. The tree is composed of different nested coalitions, in which players are part of these coalitions depending on where in the tree they are located. For the purpose of this research, we ignore any attached “support companies" with their own parallel but separate command, and instead focus on the main tree of standard infantry. We do this because this main tree is the largest and most complex.

3.3. Tree Structure

The billet hierarchy tree is composed of the following nested coalitions:

A *Fireteam* is a small team of infantrymen (players) and is the lowest in the hierarchy at the leaf nodes of the tree. A Fireteam is a strict unit composition that must adhere to the following criteria to be considered valid:

1. Must be made of exactly 4 players.[2]
2. The Fireteam consists of four roles, of which each member must have the proper qualifications to be assigned the role. These roles are:
    a) Team Leader (TL)
    b) Autorifleman (AR)
    c) Grenadier (GRN)
    d) Rifleman (RFL)
3. The Team Leader is a special role, and requires players to have a high enough rank (e.g. Corporal or Sergeant) in order to occupy.

A *Squad* is the next level above a Fireteam. Squads are composed of a Squad Leader and exactly 2 distinct Fireteams. The Squad Leader must have a higher minimum rank and additional leadership qualifications.

A *Platoon* is the next level above a Squad. Platoons are led by a Platoon Leader and a Platoon Sergeant, the latter acts as the second-in-command to the former. Both must have additional rank and training qualifications. Platoons are themselves composed of anywhere from 1 to 4 Squads, offering some flexibility to the command structure.

At the highest level is the *Company*, which consists of the Company Leader, and they are in charge of all Platoons under their command. In practice, there is little reason to go higher than this as ArmA 3 servers usually do not have enough player slots to necessitate larger structures, but actual NATO unit organization can be expanded much further if needed.

3.4. Ranks

A player agent will have a rank that represents their time, experience, and recognition of merit within a community. The ranks that we use in this paper are based on the list of ranks used by the United States Army. Each role within the hierarchy tree has a minimum rank requirement, and players cannot be assigned a role outside their rank. Players usually are assigned to the highest-ranking role that they are qualified for in order that individuals best qualified to lead may do so. However, players of a higher rank can choose to move down the hierarchy tree. This is an action we call *self-demotion*, and happens in some uncommon circumstances to fill in gaps lower in the tree, or to fulfill a player’s preferences if they would rather not lead for the day.

3.5. Qualifications

Each role within the hierarchy tree has its own set of skill or training qualifications that a player must have in order to be assigned that role. For example, an Autorifleman must have the training necessary to use their machine gun, and a Team Leader must have some basic leadership training. These qualifications are tracked for every player, and a player must possess all of the prerequisite qualifications in order to be assigned to a role. These qualifications are Boolean, as players will either have a given qualification or they will not.

Many ArmA 3 realistic MilSim communities have a large number of available qualifications to train for, many of which require prerequisite qualifications themselves. In general, the leadership of many ArmA communities elect to have simple skill trees with no more than one prerequisite per qualification whenever possible. This is to keep progression simple for players to understand and for leadership to keep track of. Our algorithms allow for the dynamic creation of a qualification tree with any number of prerequisites to explore more complicated assignments.

Gaming-based or game-inspired coalition formation games has been explored. We mention a few here, from which AACFGs take elements. The first are Anchored Team Formation Games [3, 4], which (a) assigns players to coalitions to play TTRPGs, and (b) also makes use of roles (GM and players). More relevant, perhaps, are Role Based Hedonic Games [5, 6], introduced to model preferences of League of Legends players, in which players express preferences

[1] The official ArmA 3 community website, units.arma3.com, allows communities to identify with the tags of “realism", “semi-realism," and “relaxed." The latter of these two are much less likely to have a rigid billet structure, so we will not consider them for this paper.

[2] In practice, team sizes can vary to account for different organizations or numbers of players, but for this paper we are only interested in the case where team sizes and roles are fixed.

over their own champion (role) and the champions that make up the team. In addition, Tiered Coalition Formation Games (TCFGs) [7], introduced to model the stratification of Pokémon by the fan site Smogon, has some of the flavor of the role hierarchy of AACFGs. We later give a formal definition of TCFGs and of the related notion of socially conscious stability, as we build on that stability notion.

In summary: *each player has a maximum rank they can be assigned, and a set of qualifications that enable particular billets. Each player has utilities for each other player, though the inter-player utility within a given billet is scaled so that it decreases geometrically with the branch distance in the hierarchy tree. Players on separate branches do not contribute to each other's utility.*

Armed Assault Coalition Formation Games (AACFGs) are similar to *Anchored Team Formation Games* (ATFGs), which were designed to capture table formation for table-top role-playing games [3, 4]. AACFGs and ATFGs are both extensions of *Additvely Separable Hedonic Games,* in which each agent has utility assigned to every other agent. The agent's total utility for a coalition is the sum of the assigned utilities to each other member of that coalition.

Definition 1. *[1] **Additively Separable Hedonic Games (ASHG)** are a class of hedonic game where each player $i \in N$ assigns values to each player $j \in N$, expressed as $v_i(j)$; for all i, $v_i(i)$ is 0. The utility a player a_i derives from each coalition $S \in \mathcal{N}_i$ such that $i \in S$ is defined as $u_i(S) = \sum_{j \in S} v_i(j)$.*

An assignment of agents (or here, players) to coalitions is *stable* if there is no agent or agents that would have higher utility in coalition(s) other than those assigned. Notions of stability in coalition formation games [8] vary in the number of agents that can move, which agents' utilities are considered, and whether those utilities have to be strictly improved or merely non-decreasing. The two most common notions of stability are Nash and core stability. The simplest notion, *Nash Stability* [**?**], holds if no single agent can increase its utility by joining a different coalition than the one to which it is assigned. An assignment is *core stable* [1] if no group of agents can create a new coalition of their own such that each agent in the new coalition improves their utility.

Note that Nash stability is not meaningful in games in which coalitions have fixed sizes, or have structures driven by fixed positions and roles; in such settings, an individual agent typically cannot deviate to or from a coalition without invalidating that coalition. Similarly, core stability has limited usefulness for settings with fixed positions and roles, as there may be valid core deviations under the definition of core stability which would cause agents outside of those deviations to no longer belong to a legitimate coalition. A related game, *Roles and Teams Hedonic Games,* models League of Legends teams of fixed size and considers swaps of agents between coalitions as the driver of stability [5, 9].

Swap exchange stability has a rich history, well surveyed in [10], for instance. We mention a few additional applications: kidney exchanges [11] (there is a huge literature there), in which biologically incompatible donor-recipient pairs are placed in a cycle or chain[3] where donors are swapped so recipients receive compatible kidneys; marriage markets where mates are similarly exchanged, perhaps in sequence [12]; other matching markets [13]; roommate assignment problems [14, 12, 15].

[3]Kidney exchange chains start with a kidney donor who has died.

We now formally define AACFGs. Informally, an instance consists of the baseline preferences agents have over each other, and the factors that define agents' utilities for given assignments (or billets — we use the terms interchangeably), namely the structure of the hierarchy, or tree of roles. We also include the requirements of each role, namely qualifications.

Definition 2. *An instance of an **ArmA Coalition Formation Game (AACFG)** consists of*

- *a preference matrix P that implicitly defines the number of players;*
- *an array of qualifications for those players;*
- *a discount $1/s$ for utilities as a function of distance in the assignment tree, or billet;*
- *the structure of the assignment tree (depth of the tree, qualifications for each role).*

We are interested in assignments, or *billets*, for AACFGs.

Definition 3. *A **billet** is a mapping from players to roles. A **valid billet** is one in which the assigned players each have the qualifications necessary for their assigned role.*

The utility for player a for billet B depends on a's utility for its descendants, ancestors, and (if a is assigned to a Fireteam) those in its Fireteam. We define $D^B(a)$ to be those players, and

$$u_a(B) = \sum_{c \in D_a} (B) x_a(c).$$

We define $u(B) = \Sigma_a u_a(B)$.

We borrow heavily from the work on Tiered Coalition Formation Games to define our notion of stability.

Definition 4. *[7] A* tiered coalition formation game (TCFG) *is a coalition formation game $(N, \succeq)$, where $N = \{a_1, a_2, ..., a_n\}$; an outcome, or* tier list*, is a totally ordered spanning set of disjoint coalitions $[T_1, T_2, ..., T_k]$; $\text{Seen}(a_i, T)$ for tier list T denotes the set of all agents that are in the same tier as a_i or are in a lower (prior) tier; and the preferences for each agent a_i in each possible tier list T are determined solely by the set of agents a_i "sees" in T [7].*

Note that TCFGs are not hedonic, in that an agent affected by the assignment of all seen agents, even those not in its own tier. Thus, when an agent moves from one tier to another, it potentially affects the utilities of agents in any intervening tiers.

We are interested in billets that are not only valid, but stable. We consider two notions of stability based on *swaps.*

Definition 5. *Given an instance I of AACFG and valid billet B for I, B is* swap stable *if there is no pair of agents a, b that are assigned to different roles by B, such that swapping a and b leaves the billet valid, and improves the utilities for both.*

The notion of swap stability is appealing, but may not be achievable.

Lemma 1. *Not all instances of AACFGs have swap stable billets.*

Proof. We define an instance $I = P, Q, s$ of an AACFG, and a valid billet B. Let all players have all qualifications for Fireteam roles, and let there be sufficient other qualified players to have a valid billet. Let a and b have no qualifications for higher ranks. We define the preference matrix P

by $P[a,b] = 5$ and $P[b,a] = -5$. For all other players x, $P[x,a] = 1$ and $P[x,b] = -1$, and otherwise $P[x,y] = 0$. Thus, a wants to be in a Fireteam with b, but b does not want to be in a Fireteam with a. If a and b are billeted in the same Fireteam, and any player billeted in another Fireteam (with utility 0) can improve their utility by swapping with b. If a and b are in separate Fireteams, any player in b's Fireteam (with utility -1) can improve their utility by swapping with a. Thus, a will chase b and b will run away, ad infinitum. □

The problem that arises in our counterexample is that a is allowed to join b's Fireteam, to the detriment of b's utility. There are a variety of notions of stability that extend consideration of utility beyond that of the blocking agents. To help us think about this, we define a notion of stability particular to TCFGs.

Definition 6. *A* socially consciously stable tier list *is a tier list T in which there exists no agent a_i that can improve its utility by moving to a new position in the tier list without decreasing the total utility of all other agents [16, 17].*

We can understand socially conscious stability (SCS) to require permission from all affected agents before an agent can move. It generalizes the notions of contractual stability (which requires that an agent leaving a coalition must leave behind other agents whose utility is not lowered — "permission to leave") and individual stability (which requires that an agent joining a coalition cannot lower the utility of any agent in that coalition — "permission to enter") [2]. We define a notion of stability that is similar to contractually individual stability, but is defined in terms of agents trading coalitions, or *swapping*.

Definition 7. *An assignment B is* Socially Conscious Swap Stable (SCSS) *if there is no pair of agents a and b assigned to roles $B(a)$, $B(b)$ such that swapping them creates a new assignment B' with*

- $u_a^{B'} > u_a^B$;
- $u_b^{B'} > u_b^B$;
- $\forall x \notin \{a,b\}\ u_x^{B'} \geq u_x^B$.

3.6. Computational Complexity of AACFGs

We note that AACFGs generalize Additively Separable Hedonic Games with Fixed-Size Coalitions (AS-HGFSCs) [10]. In particular, given an AS-HGFSCs, we can construct an AACFG instance by adding players and assigning the unique roles above the Fireteam level, while treating the input agents from the AS-HGPSC instance as players fully qualified for Fireteams. (Note that we need to loosen our restriction of Fireteams having exactly four members.)

From this observation, we get all of the hardness results for AA-HGFSCs and some inspiration for algorithms. For instance, in what Bilo, et al. [10] call SSS (strictly swap stability) and what we call SCSS, there is always an SCSS billet. For our work, this assumes that there is some valid billet. In the current formulation, it is easy to check for a valid billet, so we can limit our attention to instances that have them. However, for their version of the problem (which fixes the number of coalitions rather than the size, but can also specify the sizes of those coalitions), finding the socially-optimal assignment of maximal utility is NP-hard.

4. Heuristic Algorithms

We used local search and simulated annealing. Note that local search, also know as "hill climbing" [18], can only take improving moves, which tends to get stuck in local, non-global optima. Simulated annealing allows for occasional random "jumps," which have the possibility of jumping away from sub-optimal hills. Given that the range of values of pairwise utilities is polynomially bounded in the number of agents, there is a polynomial upper bound on the utility of an assignment. Note that, for SCSS, each local improvement from the local search strictly increases the social welfare (total utility). Furthermore, finding an improving swap, if one exists, takes polynomial time. Thus, each run of local search has a polynomial time bound. For simulated annealing, we used a pre-set cut-off to deal with the possibility of super-long runs.

In the simulated annealing, we used the parameter k to determine the probability a random swap is accepted weighted by a temperature value T. T decays from 1 to .001 by $T' = T/1.001$. This probability is given by

$$P(u(B), u(B'), T) = e^{\frac{u(B')-u(B)}{k*T}}.$$

Note that the problem of finding valid billets, ones where all agents have the qualifications for their assigned roles, is nontrivial, though we can easily find one such billet. For the local search and simulated annealing algorithms, we start all runs from the same initial assignment. The simulated annealing variants use random number generators, so each run may find a different stable assignment. However, local search is deterministic, thus it was only run once.

5. Experiments

5.1. Generator

The generator consists of n player structures, each with their own rank stored as a uniformly-random integer using the std::rand() function. For the skill qualifications of each player, each agent runs a series of coinflips against a global list of qualifications. Each qualification in the list requires the agent to possess all of the prior qualifications. We kept the size of the qualifications small, allowing each agent to hold anywhere from zero to five qualifications.

5.2. Runs

Twenty 50-player structures were generated. The discount value s is set to 2 and the utility one player has for another is in the interval [0,1]. For each instance, we checked if a valid billet exists by iteratively assigning the highest ranked player to the highest remaining position, irrespective of inter-player preferences. From this initial billet we ran the deterministic local search, then 500 iterations of simulated annealing with $k = 10$ and simulated annealing with $k = 1$.

6. Results

For all instances we were able to find stable assignments. Tables 1, 2, and 3 show the results from running 500 iterations of each algorithm on the 20 instances. The mean values of the $k = 10$ and $k = 1$ simulated annealing algorithms were significantly different than the local search values ($p = 4.368e^{-5}$, $p = 3.893e^{-5}$).

We observe in Table 1 that the local search algorithm falls slightly short of the average utility found through the simulated annealing algorithms in Tables 2 and 3. This is expected, as the local search executes in a single pass, and thus does not necessarily find a global optimum. This is a fair trade-off compared to simulated annealing, however, as local search is significantly faster, particularly if it's only run once.

We also observe that the simulated annealing algorithm does not significantly vary for different values of k. We see in Table 4 that the simulated annealing algorithms for $k = 10$ and $k = 1$ performed extremely similarly across 100 instances, with a very small improvement for $k = 10$, but not enough that we believe to be statistically significant.

Table 1
Local Search

Instance	Value
1	112.252
2	114.603
3	116.402
4	115.416
5	117.228
6	115.788
7	115.678
8	111.833
9	115.508
10	114.291
11	113.775
12	114.92
13	115.663
14	116.937
15	118.35
16	116.643
17	116.479
18	114.154
19	117.611
20	119.229
-	115.638

Table 1 shows the results from running local search on every instance. The value is the total utility of the billet after the local search algorithm converges to a local maximum

7. Toy Example

One question that is not answered in the previous section is whether the maximum-utility billets found by simulated annealing are actually optimal. To test this, we generated a 21-agent instance with an initial billet utility of 34.906. Then, using a branch and bound algorithm, we were able to compute the global maximum of 41.153, which ran in 3018.43 seconds and checked 24,567,994 nodes.

The local search algorithm converged at a value of 39.826 after seven swaps. Afterwards, we ran 100 iterations of simulated annealing k=10 and k=1. Given the global maximum, we can conclude that the local search converged at a local but not global maximum, while the simulated annealing algorithms were able to find the global maximum (see Table 4) in a portion of the runs (G Max(%)).

Although the mean value of the $k = 10$ algorithm is slightly higher than the $k = 1$ algorithm, there was no significant difference between the two algorithms in terms of their utility ($p = 0.784$) We also observe that the lowest utility found by either of the simulated annealing algorithms was the highest value found by the standard local search algorithm. This is unsurprising as a smaller instance size would make the search space for simulated annealing smaller.

Table 2
Simulated Annealing (k = 10)

Instance	Min	Max	Mean	Std Dev
1	111.286	118.065	114.748	1.032
2	112.962	119.673	116.303	1.105
3	113.826	119.556	116.825	1.066
4	112.885	119.241	116.138	1.065
5	115.544	121.312	118.327	1.056
6	115.212	122.779	119.589	1.283
7	114.306	120.111	117.15	1.05
8	108.672	113.863	111.156	0.954
9	114.477	120.098	117.485	1.089
10	113.471	120.537	117.453	1.064
11	112.601	118.827	115.579	1.12
12	111.261	118.539	115.899	1.038
13	114.528	121.563	118.374	1.33
14	114.71	121.152	117.759	1.121
15	114.721	121.833	118.512	1.125
16	114.38	121.413	117.723	1.031
17	116.094	121.983	119.206	0.984
18	114.061	120.354	117.346	1.06
19	116.305	122.427	119.51	1.143
20	114.041	121.191	118.102	1.137
-	113.767	120.226	117.159	1.093

Table 2 shows the results from running 500 iterations of the k=10 simulated annealing algorithm. The columns from left to right represent the instance number, the local maximum with the lowest total utility, the local maximum with the highest utility, the mean utility, and the standard deviation of the local maxima for each instance.

Table 3
Simulated Annealing (k = 1)

Instance	Min	Max	Mean	Std Dev
1	111.132	117.636	114.859	1.07
2	113.49	119.131	116.409	1.016
3	113.25	119.6	116.831	1.054
4	113.108	119.272	116.043	1.034
5	115.327	121.287	118.331	1.043
6	115.249	122.797	119.399	1.387
7	113.73	119.99	117.182	1.028
8	108.302	113.57	111.138	0.952
9	114.604	120.547	117.462	1.048
10	114.606	120.641	117.416	1.084
11	112.738	118.57	115.632	1.106
12	112.906	118.735	115.946	1.071
13	114.543	122.06	118.374	1.28
14	114.53	120.506	117.782	1.056
15	115.226	122.188	118.451	1.196
16	114.618	120.463	117.655	0.943
17	116.3	122.001	119.346	0.993
18	114.582	120.461	117.423	1.02
19	116.326	123.495	119.65	1.181
20	115.126	121.122	118.179	1.017
-	113.985	120.204	117.175	1.079

Table 3 shows the results from running 500 iterations of the k=1 simulated annealing algorithm. The columns from left to right represent the instance number, the local maximum with the lowest total utility, the local maximum with the highest utility, the mean utility, and the standard deviation of the local maxima for each instance.

8. Conclusions

ArmA 3 is a popular MilSim that requires a manual billeting process, which involves assigning 50–100 players into a set

Table 4
Toy Example Simulated Annealing

k	Min	Max	Mean	G Max(%)
10	39.699	41.153	40.844	21
1	39.826	41.153	40.83	17

Table 4 shows the results from running 100 iterations of simulated annealing. k represents the k value used for the accept function. G Max is the number of times the algorithm found the global maximum value out of 100.

of roles before the game begins. This process can be quite time consuming, and often involves much trial and error. In this paper we explore how this problem can be cast as a coalition formation problem, which enables the use of coalition formation algorithms to perform billeting automatically. Specifically, we show how simulated annealing and local search can be used to generate stable billets in polynomial time.

To our knowledge, this is the first example of using coalition formation algorithms to automate role assignment in a hierarchical game. We were able to show that simple heuristics do surprisingly well. Future work will entail scaling up the methods and developing both game-specific and general methods for billeting, and related activities.

References

[1] S. Banerjee, H. Konishi, T. Sonmez, Core in a simple coalition formation game, Social Choice and Welfare 18 (2001) 135–153.

[2] A. Bogomolnaia, M. O. Jackson, The stability of hedonic coalition structures, Games and Economic Behavior 38 (2002) 201–230.

[3] J. Schlueter, C. Addington, J. Goldsmith, Anchored team formation games, in: The International FLAIRS Conference Proceedings, volume 34, 2021.

[4] J. Schlueter, Novel Hedonic Games and Stability Notions, Ph.D. thesis, University of Kentucky, 2021.

[5] M. Spradling, J. Goldsmith, X. Liu, C. Dadi, Z. Li, Roles and teams hedonic game, in: International Conference on Algorithmic DecisionTheory, Springer, 2013, pp. 351–362.

[6] M. Spradling, Role Based Hedonic Games, Ph.D. thesis, University of Kentucky, 2015.

[7] C. Siler, Tiered coalition formation games, The International FLAIRS Conference Proceedings 30 (2017) 210–214.

[8] R. J. Aumann, J. H. Dreze, Cooperative games with coalition structures, International Journal of game theory 3 (1974) 217–237.

[9] M. Spradling, J. Goldsmith, Stability in role based hedonic games., The International FLAIRS Conference Proceedings 28 (2015) 85–90.

[10] V. Bilo, G. Monaco, L. Moscardelli, Hedonic games with fixed-size coalitions, in: AAAI Conference on Artificial Intelligence, volume 36(9), 2022, pp. 9287–9295. doi:https://doi.org/10.1609/aaai.v36i9.21156.

[11] A. E. Roth, T. Sönmez, M. U. Ünver, Kidney exchange, The Quarterly journal of economics 119 (2004) 457–488.

[12] K. Cechlárová, D. F. Manlove, The exchange-stable marriage problem, Discrete Applied Mathematics 152 (2005) 109–122.

[13] J. Alcalde, Exchange-proofness or divorce-proofness?, Economic Design (1995) 275––287.

[14] K. Cechlárová, On the complexity of exchange-stable roommates, Discrete Applied Mathematics 116 (2002) 279–287.

[15] J. Chen, A. Chmurovic, F. Jogl, M. Sorge, On (coalitional) exchange-stable matching, in: Algorithmic Game Theory: 14th International Symposium, SAGT 2021, Aarhus, Denmark, September 21–24, 2021, Proceedings 14, Springer, 2021, pp. 205–220.

[16] N. Arnold, J. Goldsmith, S. Snider, Extensions to tiered coalition formation games, The International FLAIRS Conference Proceedings 35 (2022). doi:https://doi.org/10.32473/flairs.v35i.130708.

[17] N. Arnold, S. Snider, J. Goldsmith, Socially conscious stability for tiered coalition formation games, Annals of Math and Artificial Intelligence (2024). https://link.springer.com/article/10.1007/s10472-023-09897-4.

[18] C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and complexity, Courier Corporation, 2013.